Ćwiczenia z tablic i obiektów

– wykonał: Łukasz Susłowicz

1. Stwórz funkcję mergeArrays(array1, array2) , która zwróci nową tablicę, będącą połączeniem dwóch przekazanych tablic.

```
JS tablice_i_obiekty.js > ...
1   const array1 = [2, 5];
2   const array2 = [3, 7];
3
4   const bothArrays = [
5       ...array1,
6       ...array2,
7   ];
8
9   const mergeArrays = () => bothArrays;
10  mergeArrays([2, 5], [3, 7]);
```

```
> mergeArrays()
< ▶ (4) [2, 5, 3, 7]
>
```

2. Stwórz funkcję exceptFirst() , która zwróci tablicę ze wszystkimi przekazanymi do niej argumentami oprócz pierwszego.

```
JS tablice_i_obiekty.js > ...
1   const tab = [2, "delfin", null];
2   const [first, ...restParam] = [2, "delfin", null];
3   const exceptFirst = (tab) => {
4   console.log(restParam);
5   };
6   exceptFirst();
```

```
▶ (2) ['delfin', null]
>
```

3. Stwórz funkcję last2Parameters() , która zwróci tablicę z dwoma ostatnimi argumentami przekazanymi do funkcji.

```
JS tablice_i_obiekty.js > ...
1   const arrayEx3 = [1, 3, undefined, true];
2   const newOne = arrayEx3.slice(-2);
3
4   const last2Parameters = (arg1, arg2, arg3, arg4) => [arg1, arg2, arg3, arg4].slice(-2);
5   last2Parameters(1, 3, undefined, true);
```

```
> last2Parameters(1, 3, undefined, true);
< ▶ (2) [undefined, true]
>
```

4. Stwórz funkcję indexOf(element, array) , która zwróci indeks podanego elementu w podanej tablicy.

```
JS tablice_i_obiekty.js > indexOf
1   const element = "kawa";
2   const array = ["rum","kawa","sangria"];
3   const indexOf = (element, array) => {
4       return array.indexOf(element);
5   };
```

```
> indexOf("kawa", ["rum", "kawa", "sangría"])
< 1
>
```

5. Stwórz funkcję findNonEmptyTask(tasks) , która zwróci pierwsze zadanie, które ma treść.

```
JS tablice_i_obiekty.js > ...
1   const tasks = [{ content: "" }, { content: "kupić balony w kształcie psów" }];
2   const notEmptyContent = ({ content }) => content !== "";
3   const findNonEmptyTask = (tasks) => tasks.find(notEmptyContent);
4
```

```
> findNonEmptyTask([{content: ""},{content: "kupić balony w kształcie psów"}])
< ▶ {content: 'kupić balony w kształcie psów'}
>
```

6. Stwórz funkcję oddIndex(numbers) , która zwróci indeks pierwszej nieparzystej liczby z podanej tablicy.

```
JS tablice_i_obiekty.js > ...
1   const tab = [2, 4, 7, 8];
2   const isEven = number => !!(number % 2);
3   const oddIndexNotFunc = tab.findIndex(isEven);
4   const oddIndex = (numbers) => numbers.findIndex(isEven);
```

```
> oddIndex([2,4,7,8])
< 2
>
```

7. Stwórz funkcję hasStrawberry(fruits) , która sprawdzi, czy w podanych owocach jest truskawka.

```
const hasStrawberry = (fruits) => fruits.includes("truskawka");
```

```
> hasStrawberry(["banan", "marakuja"])
< false
> hasStrawberry(["truskawka"])
< true
>
```

8. Stwórz funkcję someAdult(people) , która sprawdzi, czy wśród podanych osób jest ktoś dorosły.

```
JS tablice_i_obiekty.js > ...
1   const people = [
2       { name: "Melodia", age: 15 },
3       { name: "Kosmo", age: 19 },
4   ];
5   const isSomoneAdult = ({ age }) => age >= 18;
6   const someAdult = (people) => people.some(isSomoneAdult);
```

```
> someAdult([{name: "Melodia", age: 15},
  {name: "Kosmo", age: 19}])
< true
>
```

9. Stwórz funkcję onlyString() , która sprawdzi, czy wszystkie podane argumenty są tekstowe.

```
JS tablice_i_obiekty.js > ...
1   const numbers = [1, "żelazko"];
2
3   const isString = variable => {
4       if (typeof variable === "string") {
5       return true;
6       }
7       else {
8           return false;
9       }
10  };
11
12  const onlyString = (numbers) => numbers.every(isString);
13
```

```
> onlyString(["","żelazko"])
< true
>
```

#W pytaniu chyba jest błąd, bo dotychczas parametrem funkcji była tablica a tu nie jest.

10. Stwórz funkcję filterPremium(carBrands) , która z podanej tablicy marek samochodów zwróci tylko marki premium.

```
JS tablice_i_obiekty.js > ...
1   const regularCars = ["Peugeot", "BMW", "Audi"];
2
3   const premium = ["BMW", "Audi", "Mercedes"];
4
5   const isPremium = (cars) => (cars.includes("BMW") || cars.includes("Audi") || cars.includes("Mercedes"));
6
7   const filterPremium = (cars) => cars.filter(isPremium);
```

```
> filterPremium(["Peugeot", "BMW", "Audi"])
< ▶ (2) ['BMW', 'Audi']
>
```

11. Stwórz funkcję getColors() , która zwróci tablicę kolorów samochodów podanych w argumentach.

```
JS tablice_i_obiekty.js > ...
1   const colors = [
2       { car: "BMW X5", color: "black" },
3       { car: "Peugeot 3008", color: "white" },
4   ];
5   const getColors = (colors) => colors.map(({ color }) => color);
```

```
> getColors([
      { car: "BMW X5", color: "black" },
      { car: "Peugeot 3008", color: "white" },
  ])
< ▶ (2) ['black', 'white']
```

12. Stwórz funkcję sortPeople(people) , która posortuje osoby od najmłodszych do najstarszych.

```
JS tablice_i_obiekty.js > ...
1   const people = [
2       { name: "Krzychu", age: 30 },
3       { name: "Zdzichu", age: 20 },
4   ];
5   //const sorts = ((a, b) => a - b);
6   const sortPeople = (people) => people.sort(({ age }) => -age);
```

```
> sortPeople([
      { name: "Krzychu", age: 30 },
      { name: "Zdzichu", age: 20 },
  ])
< ▼ (2) [{…}, {…}]
    ▶ 0: {name: 'Zdzichu', age: 20}
    ▶ 1: {name: 'Krzychu', age: 30}
      length: 2
    ▶ [[Prototype]]: Array(0)
>
```